Computing Machinery I
Project
(40% of your final score)

Lead TA
The lead TA for this project is Abdelghani (`abdelghani.guerbas@ucalgary.ca`).

Objective
You will create a simple single-player game both in C and ARM assembly. This is a two-part project.

Overview
This game is inspired by the retro **bomberman** game, but it is an oversimplified version. The game consists of a 2D board of hidden rewards and scores. The scores are represented as floating point numbers, positive and negative. The board tiles are randomly populated by these numbers. In addition, the board has an *exit* tile and reward tiles that double the range of a bomb. When the user uncovers the exit tile, the game is won. The player starts with three lives and a score of zero and a specific number of bombs. S/He chooses a cell to place a bomb. The bomb explodes uncovering the immediate bordering tiles to the bomb. The user's score is updated as the sum of all the values of the uncovered tiles. If the score becomes negative a life is lost and the number of lives is decremented. If a reward tile is uncovered, the bomb range is doubled: it will be able to uncover the immediately surrounding tiles and their immediately surrounding tiles. If n reward tiles (n $) were found, 2^n layers would be uncovered. Unlike the original game, the bomb never kills the bomberman. The game ends in one of three cases: when the score becomes less than or equal to zero and the lives are zeroed, when the player uncovers the exit tile on the board, or the total number of bombs is exhausted before uncovering the exit tile. When the user starts a new life, the total number of bombs is maintained from the previous life and the score is zeroed. There is no need for a graphical interface.

Details
The user starts the game by specifying a player name and the dimensions of the game board. These will be N×M grids. The minimum value of N and M is 10 and there is no maximum, as long as the game board can be fit on the screen. For example, the following command line invocation starts the game with a 100x200 grid:

```
./mygame.o bomberman 100 200
```

The grid is then initialized with random floating numbers with two precision points, ensuring that around 40% of the cells are negative, around 40% of cells are positive and the remaining cells, around 20% are bonus pack tiles including the exit tile '*' . These numbers should be non-zero with a maximum absolute value of 15. You must use bitwise arithmetic to calculate the modulus of the random numbers.

To help with grading, display the uncovered game board, showing all the contents of the tiles and also stating the ratio of negative tiles. Therefore, if the board is 3x4 (Remember: the minimum value of N and M is 10 and there is no maximum) your program should display something like the following before the actual game starts:

```
  1.00  -2.10     $     0.50
 14.00  -7.20  -1.10   12.00
 -0.70     *   5.00    -4.01
```

Negative numbers 5/12 = 41.67%
Positive numbers 5/12 = 41.67%
Special tiles 2/12 = 16.67%

The normal (covered) game board is then displayed. We will give an example using 5×5 grid (such a board size is too small to result in a meaningful game). The initial screen will look like the following (note that the number of bombs needs to be appropriate for the game board size):

```
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
Lives: 3
Score: 0
Bombs: 3
```

Prompt the user to choose a tile to place the bomb by entering its coordinates, such as:

```
Enter bomb position (x, y): 0 1
```

The move is applied, and the resulting game state is shown again (+ indicates a positive score and – a negative score). For example:

```
Total uncovered score of 5.23 points
++-XX
-++XX
XXXXX
XXXXX
XXXXX
Lives: 3
Score: 5.34
Bombs: 2
```

A bomb-range-doubled reward is represented by a dollar ($) symbol. For example, if tile 1 1 contains such a reward, the response to placing the bomb in tile 0 1 above may look like:

```
Bang!! Your bomb range is doubled
Total uncovered score of 5.23 points
++-XX
-$+XX
XXXXX
XXXXX
XXXXX
Lives: 3
Score: 5.34
Bombs: 2
```

Since a $ is uncovered, the next bomb will have a double-range effect. This effect lasts for one bomb only. The exit tile is represented using the * symbol.

In addition, the game may utilize other surprise packs. These will be given bonus points and it is left to your creativity to choose such packs. No bonus points will be given to surprise pack if the double-range-effect feature is not implemented.

The user can choose at any time to exit the game. When the game ends with a win or a loss, the player's name, score, and duration of the play are recorded in a log file. A user can also ask to display the top *n* scores before or after any game, including player names and duration.

Modularity
Your code must be divided into functions as appropriate. At a minimum, you must define the following functions (we are not showing all necessary arguments):

- `initializeGame(*board)`
- `randomNum(n,m,neg);` `n` and `m` are the lower and upper bounds for the random number; the generated number is negative if `neg` is `true`.
- `displayGame(*board)`
- `calculateScore(),` returns overall score
- `logScore()`
- `exitGame()`
- `displayTopScores(n)`

Part 1 (20% of your final score, due on October 21st @ 11:59 MST)
Create the game entirely in C.
Note: Your C code must run on the university servers.

Part 2 (20%, of your final score, due on December 09th @ 11:59 MST)
Re-create the same game entirely in ARMv8 assembly. For this part, we suggest that you implement the game first without floating-point numbers. Then, you can add this feature at the end of the semester. If your code is modular enough. It should only require you to replace a few subroutines.

Submission
Submit your work to the appropriate dropbox on D2L. The lead TA will provide further submission instructions.

Late Submission Policy
Late submissions will be penalized as follows:
-12.5% for each late day or portion of a day for the first two days
-25% for each additional day or portion of a day after the first two days
Hence, no submissions will be accepted after 5 days (including weekend days) of the announced deadline.

Academic Misconduct
This project is to be done by individual students: your final submission must be your own original work. Teamwork is not allowed. Any similarities between submissions will be further investigated for academic misconduct. While you are encouraged to discuss the project with your colleagues, this must be limited to conceptual and design decisions. Code sharing by any means is prohibited, including *looking* at someone else's paper or screen. The submission of compiler generated assembly code is absolutely prohibited. Any re-used code of excess of 5 lines in C and 10 lines in assembly (10 assembly language instructions) must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.

D2L Marks
Marks posted on D2L are subject to change (up or down).

C- Requirement
As per the course syllabus, a cumulative C- must be achieved in the project in order to achieve a C- in the course.

Computing Machinery I
Project Rubric

Student:_____

Item	Max Points	Points
Code compiles	5	
Program displays uncovered board and percentage of negative numbers	5	
Game logic	30	
Use of floating-point numbers	10	
User interface (input validation, implementing all features)	10	
Random numbers	10	
Use of binary arithmetic (mod with random numbers)	5	
Modularity and macros (macros: part 2)	10	
Command-line arguments	5	
Passing array parameters by reference	5	
Code readability (documentation)	5	
Bonus (surprise packs)	**10 (bonus)**	
Total Points	**100**	